BAB IV. ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisa Kebutuhan Sistem

Analisis kebutuhan digunakan untuk mengetahui apa saja yang akan digunakan dalam proses perancangan sistem. Dalam membangun sistem aplikasi ini, diperlukan analisis kebutuhan yang jelas sebagai tujuan utamanya agar tidak keluar dari rencana yang telah ditetapkan. Berdasarkan hasil analisa terhadap kebutuhan dalam penelitian ini dibutuhkan *hardware* dan *software*. Kebutuhan *hardware* dan kebutuhan *software* dapat dilihat pada Tabel 4.1 sedangkan kebutuhan *hardware* dapat dilihat pada Tabel 4.2 dan kebutuhan *software* dapat dilihat pada Tabel 4.3

Tabel 4. 1. Kebutuhuan *Hardware* dan *Software*

Hardware	*Software*	Bahasa Pemrograman
Laptop	Android Studio	Java
Smartphone		

a. Kebutuhan *Hardware*

Spesifikasi *Hardware* yang digunakan dalam melakukan penelitian ini yakni :

Tabel 4. 2. Kebutuhan *Hardware*

No.	*Hardware*	Keterangan
1.	Laptop	*Processor*: Intel Core i5-7200U RAM: 12 GB VGA: NVIDIA GeForce 930MX
2.	*Smartphone*	*Phone*: Xiaomi Redmi Note 8 *Camera*: 48 MP

b. Kebutuhan *Software*

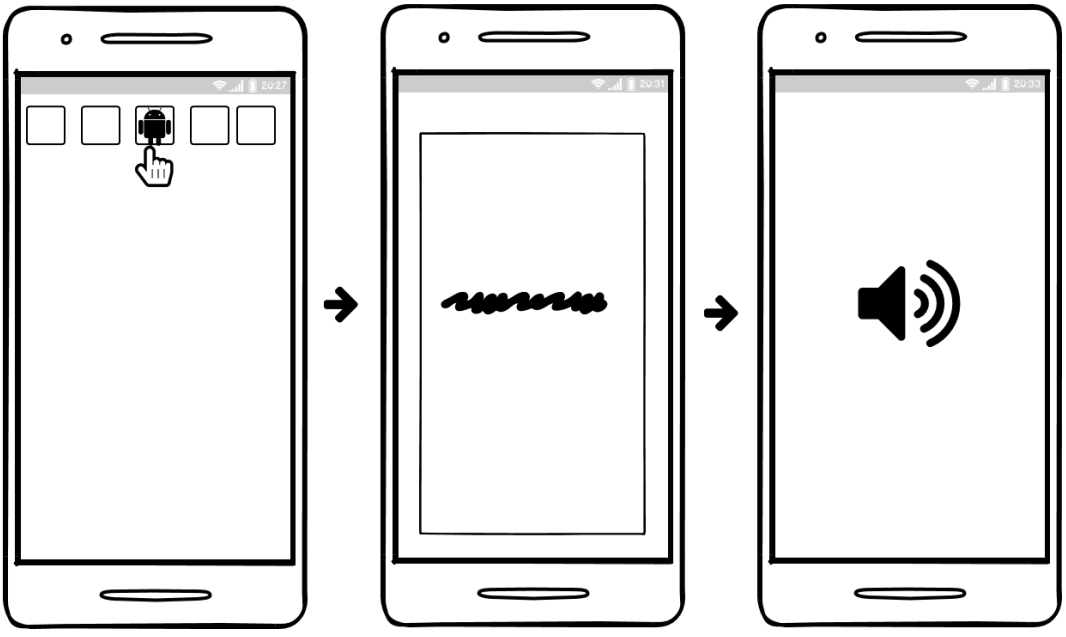
Untuk dapat melakukan melakukan penelitian pada sistem pengolahan citra untuk membaca teks berbasis android menggunakan metode *template matching correlation* berikut adalah kebutuhan *Software* :

Tabel 4. 3. Kebutuhan *Software*

No.	*Software*	Keterangan
1.	Android Studio	*Software* pengembang aplikasi untuk membuat aplikasi untuk android

4.2 Desain Sistem

Desain Sistem digunakan untuk gambaran umum dalam proses pengambilan *input* data sampai *output* data yang akan dihasilkan oleh sistem. Desain Sistem dapat dilihat pada Gambar 4.1.



Gambar 4. 1 Desain Sistem.

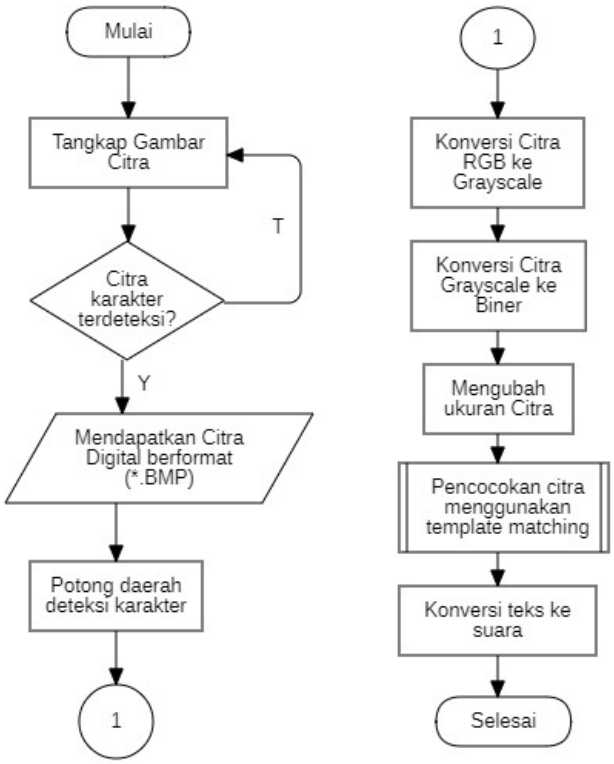
Pada Gambar 4.1 didapatkan desain sistem yaitu pertama membuka aplikasi terdahulu. Setelah aplikasi terbuka dengan sempurna, aplikasi siap digunakan untuk mendeteksi teks yang ada. Pengguna hanya mengarahkan *smartphone* ke objek

yang ingin dikenali, kamera akan mengambil citra secara otomatis. Jika kamera mendeteksi adanya teks, aplikasi akan melakukan pengolahan citra digital. Pada bagian *output* akan menampilkan hasil dari pengolahan citra berupa suara hasil teks yang dideteksi.

4.3 Perancangan Sistem

Pada tahap perancangan sistem dilakukan *input* citra gambar, proses pengolahan citra dan keluaran *output* hasil klasifikasi. Secara keseluruhan, Proses alur klasifikasi menggunakan metode *template matching correlation* dapat dilihat pada Gambar 4.2. Pada penelitian ini menggunakan *Flowchart* diagram untuk menggambarkan urutan aplikasi berjalan. Aplikasi ini terbagi menjadi dua macam *flowchart*. *Flowchart* sistem, dan metode *template matching correlation*.

4.3.1 *Flowchart* Sistem

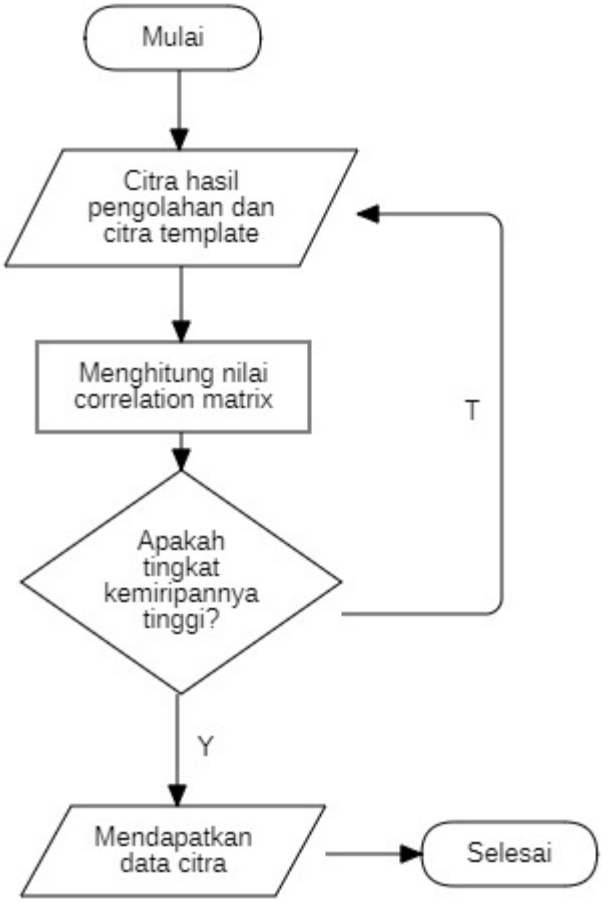


Gambar 4. 2 *Flowchart* sistem.

Pada Gambar 4. 2 *Flowchart* sistem menjelaskan bahwa langkah pertama yaitu proses tangkap gambar citra. Untuk tangkap gambar citra ini dilakukan secara otomatis, jadi tidak perlu menekan / menyentuh layar untuk menangkap gambar citra. Setelah menangkap gambar citra, citra diproses apakah ada karakter yang terdeteksi? Jika tidak, sistem melakukan tangkap gambar citra lagi. Jika iya, maka sistem mendapatkan data berupa citra digital dengan format .BMP.

Selanjutnya untuk memudahkan pemrosesan, dilakukan potong daerah deteksi karakter pada citra. Kemudian hasil dari potong daerah deteksi citra, dikonversi ke *grayscale* dan konversi ke biner. Setelah citra dikonversi menjadi biner, ukuran citra disesuaikan dengan *template*. Setelah citra diproses sistem, maka proses selanjutnya ialah tahap pencocokan citra angka dengan *template* yang sudah tersedia menggunakan metode *Template Matching Correlation*. Metode ini dapat menghitung nilai korelasi matriks citra masukan dan citra *template*. Setelah dihitung, sistem akan menunjukkan hasil *Template Matching* nya berupa audio mengenai citra teks yang dikenali.

4.3.2 Flowchart Template Matching



Gambar 4. 3 *Flowchart* pencocokan citra menggunakan *template matching.*

Pada gambar 4.3 menjelaskan proses pertama adalah citra hasil pengolahan dan citra *template*. Setelah itu sistem akan menghitung nilai *correlation matrix* nya. Hasil perhitungannya menentukan apakah memiliki tingkat kemiripan yang tinggi? Jika iya, maka data tersebut akan di simpan. Jika tidak, maka akan kembali mencocokan citra hasil pengolahan dan citra *template* selanjutnya.

4.4 Pemotongan citra

Pemotongan citra dilakukan untuk mempermudah pemrosesan huruf. Proses ini dibantu dengan *library Google Vision* untuk mencari posisi huruf pada citra dan berapa banyak huruf yang dideteksi. Sistem akan mendapatkan 4 titik koordinat yang akan membentuk sebuah persegi / persegi panjang.

Gambar 4. 4 Contoh citra diberi *marker*

Setelah mendapatkan 4 koordinat, maka dilakukan penghitungan jumlah huruf untuk satu persegi tersebut. Selanjutnya mencari panjang dari persegi tersebut dengan menggunakan rumus 4.1 berikut:

$$panjang = koordinat\ kanan - koordinat\ kiri \quad (4.1)$$

Setelah menemukan panjang, dilakukan pemotongan per huruf, dengan menggunakan rumus 4.2 berikut:

$$panjangperhuruf = \frac{panjang}{jumlah\ huruf} \quad (4.2)$$

Langkah selanjutnya yaitu memindahkan huruf hasil pemotongan tersebut ke bentuk citra baru. Langkah ini dilakukan agar citra asli tidak hilang / tidak tertindih dengan citra yang lain, yang menyebabkan hilangnya citra masukkan / citra hasil tangkapan kamera.

Contoh perhitungannya, jika diketahui nilai sisi tepi kanan adalah 290, nilai sisi tepi kiri adalah 140, dan jumlah huruf adalah 3. Maka nilai Panjang dan Panjang per hurufnya adalah:

$$panjang = 290 - 140 = 150\ pixel$$

$$panjangperhuruf = \frac{150}{3} = 50\ pixel$$

4.5 Konversi Citra ke *grayscale*

Konversi citra ke *grayscale* ini dilakukan untuk memudahkan ekstraksi huruf. Untuk mengonversi citra ke *grayscale* menggunakan rumus 4.3 berikut:

$$Y' = 0.299R' + 0.587G' + 0.114B' \quad (4.3)$$

Keterangan:

Y’ = hasil / komponen luma nonlinier

R’ = nilai pixel warna merah

G’ = nilai pixel warna hijau

B’ = nilai pixel warna biru

Contoh perhitungan, jika diketahui nilai R (*red*) = 204, G (*green*) = 193, dan B (*blue*) = 187 di pixel (45, 7), maka nilai *grayscale* nya adalah:

$$Y' = 0.299x204 + 0.587x193 + 0.114x187$$

$$Y' = 60{,}996 + 113{,}291 + 21{,}318 = 195{,}615$$

Maka untuk di citra hasil konversi grayscale di pixel (45, 7) adalah nilai R (*red*) = 195,615, G (*green*) = 195,615, dan B (*blue*) = 195,615

4.6 Konversi Citra ke biner

Setelah melakukan konversi ke grayscale, dilakukan konversi ke biner. Konversi ini membutuhkan nilai *thresholding* / nilai ambang batas. Penulis menggunakan nilai *threshold* = 100. Citra biner ini hanya memiliki 2 nilai, yaitu 255 untuk warna putih, dan 0 untuk warna hitam. Untuk mengonversi ke biner menggunakan rumus 4.4 berikut:

$$dst(x,y) = \begin{cases} 255, & if\ src(x,y) > thresh \\ 0, & otherwise \end{cases} \quad (4.4)$$

Keterangan:

dst (x, y) = citra hasil

src (x, y) = citra masukkan

(x, y) = pixel x, y

thresh = nilai *thresholding*

4.7 Mengubah ukuran citra

Langkah ini dilakukan untuk menyamakan ukuran citra yang diproses dengan ukuran citra *template*. Langkah pertama untuk mengubah ukuran citra yaitu mencari skala citra, menggunakan rumus 4.5 berikut:

$$scale = \frac{newSize}{src(height)} \quad (4.5)$$

Keterangan:

scale = skala citra

newSize = nilai ukuran baru (pixel)

src(height) = nilai tinggi dari citra yang dimasukkan (pixel)

Kemudian menentukan nilai lebar dan tinggi yang baru, menggunakan rumus 4.6 dan 4.7 berikut:

$$newWidth = src.width\ x\ scale \quad (4.6)$$

$$newHeight = scale \quad (4.7)$$

Keterangan:

newWidth = nilai lebar baru (pixel)

newHeight = nilai tinggi baru (pixel)

src.width = nilai lebar pada citra masukkan (pixel)

Contoh perhitungan, jika ukuran citra masukkan 200x200, sedangkan ukuran citra *template* 100x100, maka citra masukkan perlu diubah ukurannya menjadi 100x100. Pertama-tama mencari skala nya, dengan cara nilai ukuran baru dibagi dengan ukuran tinggi citra masukkan.

$$scale = \frac{100}{200} = 0{,}5$$

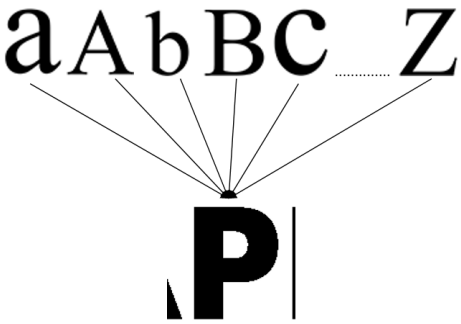
Kemudian mencari nilai lebar dan tinggi yang baru. Untuk nilai lebar didapatkan dari hasil perkalian nilai lebar pada citra masukkan dikali dengan nilai skala yang diketahui sebelumnya. Untuk nilai tinggi didapatkan dari nilai skala.

$$newWidth = 200\ x\ 0{,}5 = 100$$

$$newHeight = 100$$

4.8 Template Matching

Pada langkah ini dilakukan pencocokkan citra hasil pengolahan sebelumnya dengan semua citra *template* yang ada di sistem.



Gambar 4. 5 Ilustrasi *Template Matching*

Kesamaan antar dua buah matriks citra dapat dihitung nilainya dengan menghitung nilai korelasinya (*correlation*). Nilai korelasi dua buah matriks dapat dihitung dengan menggunakan rumus 4.8 (Hartanto et al., 2015).

$$r = \frac{\sum_{k=1}^{n}(x_{ik}-\overline{x_i}).(x_{jk}-\overline{x_j})}{\sqrt{\left[\sum_{k=1}^{n}(x_{ik}-\overline{x_i})^2.\sum_{k=1}^{n}(x_{jk}-\overline{x_j})^2\right]}} \quad (4.\ 8)$$

Dimana x_i dirumuskan dengan persamaan 4.9 dan x_j dirumuskan dengan persamaan 4.10 (Hartanto et al., 2015).

$$\overline{x_i} = \frac{1}{n}\sum_{k=1}^{n} x_{ik} \quad (4.\ 9)$$

$$\overline{x_j} = \frac{1}{n}\sum_{k=1}^{n} x_{jk} \quad (4.\ 10)$$

Keterangan:

r adalah nilai korelasi antara dua buah matriks (nilainya antara -1 dan +1).

x_{ik} adalah nilai *pixel* ke-k dalam matriks i.

x_{jk} adalah nilai *pixel* ke-k dalam matriks j.

x_i adalah rata-rata nilai *pixel* matriks i.

x_j adalah rata-rata nilai *pixel* matriks j.

n menyatakan jumlah *pixel* dalam suatu matriks.

Contoh perhitungan sederhananya, diketahui setelah menangkap sebuah citra, citra tersebut terdapat huruf "A". Setelah melakukan proses pengolahan citra, didapatkan citra biner seperti pada tabel 4.4

Tabel 4. 4. Citra masukkan berupa biner

Pixel	0	1	2	3	4	5	6	7	8	9
0										
1					1	1				
2					1	1				
3				1			1			
4				1			1			
5			1	1			1			
6		1	1	1	1	1	1	1		
7		1						1		
8		1							1	
9	1	1								1

Langkah selanjutnya melakukan pengecekan / membandingkan apakah pada pixel (x, y) di citra masukkan sama dengan pada *pixel* (x, y) di citra *template*. Contoh beberapa citra *template* pada tabel 4.5, 4.6, dan 4.7

Tabel 4. 5. Citra *template* huruf "A"

Pixel	0	1	2	3	4	5	6	7	8	9
0					1	1				
1					1	1				
2					1	1				
3				1			1			
4				1			1			
5			1	1			1	1		
6			1	1	1	1	1	1		
7			1					1		
8		1							1	
9	1	1							1	1

Tabel 4. 6. Citra *template* huruf "B"

Pixel	0	1	2	3	4	5	6	7	8	9
0		1	1	1	1	1	1			
1		1						1	1	
2		1						1	1	
3		1						1	1	
4		1	1	1	1	1	1	1		
5		1	1	1	1	1	1	1		
6		1						1	1	
7		1						1	1	
8		1						1	1	
9		1	1	1	1	1	1			

Tabel 4. 7. Citra template huruf "C"

Pixel	0	1	2	3	4	5	6	7	8	9
0				1	1	1	1			
1			1	1				1		
2		1	1							
3		1								
4		1								
5		1								
6		1								
7		1	1							
8			1	1				1		
9				1	1	1	1			

Misal jika pada *pixel* (1,1) pada citra masukkan dengan *pixel* (1,1) pada citra *template* huruf "B" sama nilai binernya, maka dianggap benar / mendapatkan poin 1. Sebaliknya, jika pada *pixel* (1,1) pada citra masukkan dengan *pixel* (1,1) pada citra *template* huruf "B" nilai biner nya tidak sama, maka mendapatkan poin 0. Dari hasil pencocokan citra masukkan pada Tabel 4.4 dengan citra *template* pada Tabel 4.5, Tabel 4.6, dan Tabel 4.7, setiap tabel dihitung nilai korelasi nya, dengan menjumlahkan poin 1 yang ada pada tabel. Hasil perhitungan nilai korelasi bisa dilihat pada tabel 4.8.

Tabel 4. 8. Hasil perhitungan nilai korelasi dengan citra *template*

Citra	Nilai
Citra masukkan dengan *template* A	93
Citra masukkan dengan *template* B	55
Citra masukkan dengan *template* C	57

Maka citra *template* yang memiliki tingkat kemiripan tinggi yaitu citra *template* huruf A.